uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías de Telecomunicación
2016/2017

Trabajo Fin de Grado

Estudio de conjuntos de clasificadores generados mediante el algoritmo class-switching

---

Adrián Vázquez Romero

Tutora: Lorena Álvarez Pérez
Fecha de presentación: 10 de julio del 2017

Índice general

Índice de figuras

Índice de tablas

Abstract

In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (also called classes) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. In this regard, a simple example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics (also known as attributes of features) of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).

An algorithm that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category or class.

Taking a further step, **ensembles of classifiers** attempt to induce a collection of diverse classifiers or predictors which are both accurate and complementary, so that, when the decisions of the different learners are combined, better prediction accuracy on previously unseen data is reached. In other words, the goal is to generate from a given training data set a collection of diverse predictors whose errors are uncorrelated. Ensembles built in this manner often exhibit significant performance improvements over a single predictor not only in many classification problems but also in regression problems. In this respect, ensembles can be built using different base classifiers: decision stumps, decision trees, neural networks, support vector machines, etc.

To create **diversity**, ensemble methods introduce perturbations at some stage in the generation of individual classifiers. These modifications, which often involve injecting some amount of randomness, can be either in the algorithm that is used to build base learners or in the training data used as input for the induction process.

The mentioned perturbations in the training data set can be introduced in different ways: Using bootstrap samples from the training data, modifying the empirical distribution of the data (either by resampling or re-weighting examples), manipulating the input features or altering the output targets and so on.

The **ensemble method** analyzed in this **Bachelor's Thesis** belongs to this last group of techniques. It is based on **randomly modifying the class label** of a fraction of samples of the training set in order to create each diverse base

classifier. This technique is commonly known as the **"class-switching"** method. In the literature, it has been shown that "class-switching" ensembles composed of a sufficiently large number of unpruned decision trees trained on data where a fairly large fraction of the class-labels are switched exhibit good performance in a large number of benchmark classification problems.

With this idea in mind, in this work, we have explored the benefits of using the technique "class-switching" when the base classifiers or base learners are based on widely used Multilayer Perceptrons (MLPs) or in the simple k-nearest neighbours (k-NN) algorithm. Furthermore, we extend the analysis of class-switching ensembles and compare their performance when two different methods of aggregating or combining the outputs values of the base learners are used: the **majority vote rule** and the **mean of soft outputs rule**. On the one hand, the vote rule operates on class labels assigned to each sample by the respective base classifiers by hardening their soft decision outputs using the maximum value selector. On the other hand, the mean of soft outputs rule, the rule directly sums the soft outputs of each individual base learners for each class hypothesis, normally delivered in terms of a posteriori class probabilities. The fused decision is obtained by applying the maximum value selector to the class dependent averages.

Experiments have been carried out on five benchmark classification problems obtained from the UCI machine learning repository and show that classification ensembles composed of MLPs can obtain significant improvements in the generalization accuracy over a single MLP-based classifiers and slightly improve the classification performance when the base learners are based on the well-known k-NN algorithm. This has the advantage that the memory requirements to store the ensemble are lower than that demanded with ensembles of k-NN classifiers.

Moreover, it has been shown experimentally that better results are always achieved when the output values of the learners are aggregated by using the "mean of soft outputs rule". This way of aggregating in the ensembles achieves the lowest possible error rate over the test set.

With this in mind, this work is organized as follows: Chapter 1 illustrates the factors that have motivated this research project, as well as a concise socio-economic analysis and the normative environment for the use of algorithms in the fields of Big Data and machine learning. Chapter 2 theoretically introduces the class-switching algorithm and the strategies that we have used to build accurate ensembles composed of MLP or k-NN classifiers. Chapter 3 describes in detail the practical implementation that we propose to design the ensembles as well as the two alternatives of aggregating the output values of the base learners explored in this work. Chapter 4 presents the results of the experiments that are compared, in terms of classification performance, with those obtained with a single classifier in five datasets. Chapter 5 describes in detail the planning of this project and its phases from the beginning (January, 2017) to its date of completion (July 10th, 2017), when this project will be defended face-to-face in an evaluation court. To complete the definition of this project, costs (both personnel and equipment) are

also displayed. Additionally, some possible indirect project costs are also depicted. Finally, Chapter 6 summarizes the conclusions of this research work and outlines some future lines.

Glosario y acrónimos

Big Data: Concepto que hace referencia a conjuntos de datos tan masivos que métodos tradicionales del procesamiento de datos no son suficientes para tratar con ellos.

Class-flipping: algoritmo basado en métodos de conjuntos que cambia las etiquetas de las muestras de entrenamiento.

CV: *cross-validation* o validación cruzada.

INE: Instituto Nacional de Estadística.

k-NN: *k-Nearest Neighbours algorithm* o algoritmo de los k vecinos más cercanos. Algoritmo de clasificación basado en distancias.

Machine-learning: (aprendizaje automático) es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas capaces de aprender a partir de datos.

MATLAB: MATrix LABoratory. Herramienta de *software* matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M).

MLP: Multilayer Perceptron o perceptrón multicapa. Tipo de red neuronal artificial capaz de resolver problemas de clasificación linealmente no separables.

TIC: Tecnologías de la Información y la Comunicación.

Toolbox: conjunto de funciones y algoritmos de cálculo especializados en un área del conocimiento que proporciona MATLAB.

UCI: University of California, Irvine o en español, Universidad de California, situada en Irvine.

Capítulo 1

Motivación y objetivos

En el presente Trabajo Fin de Grado (TFG) se explora una técnica utilizada en la literatura para introducir diversidad en un conjunto de clasificadores base conocida como "cambio de etiquetas" o, como se referirá en la memoria, con su nombre en inglés, más aceptado, *class-switching*.

En este capítulo se presentan, en primer lugar, los aspectos que han motivado la realización de este proyecto, el entorno socio-económico en que se enmarca y el marco regulador aplicable. A continuación se describen los objetivos planteados para el TFG y se presenta la estructura de la memoria.

1.1. Motivación del trabajo

Debido a la relevancia que ha tomado en los últimos años el uso de grandes cantidades de datos y sus posibilidades de análisis mediante algoritmos de *Big data* y *Machine learning* o aprendizaje automático, surge el interés por analizar, desarrollar e intentar mejorar las prestaciones de alguno de estos algoritmos.

No es un secreto cómo empresas como *Facebook* [1], gobiernos y ciudades [2], fabricantes de máquinas autónomas [3] o investigadores de diversos campos como la medicina [4] llevan años utilizando grandes volúmenes de datos generados por usuarios, sensores y aparatos conectados a la red para mejorar, ganar dinero, ayudar a los ciudadanos... Aún así, las investigaciones en este campo aún no han llegado a su cima y a día de hoy existe una parte importante del presupuesto de gobiernos, universidades y empresas dedicados a buscar y utilizar algoritmos basados en el tratamiento de estos datos.

Gran parte de estos algoritmos son los dedicados al tratamiento de muestras diferenciadas en clases o grupos, como pueden ser la separación en grupos de individuos con diferentes atributos en común, los problemas de diagnóstico médico biclase, los algoritmos de recomendación de música o libros, etc. La mayoría de estas técnicas utilizan muestras de las cuales se conocen tanto sus características $\mathbf{x}$ como la clase y a la que pertenecen para así, tras diseñar el clasificador de forma correcta, cuando se reciban únicamente los atributos de un nuevo candidato sea posible etiquetar dicho ejemplo dentro de una de las clases conocidas, y, si se desea,

con un nivel de confianza asociado. Como es lógico, estos clasificadores cometen errores al ofrecer sus resultados. Es tarea del diseñador lograr que los algoritmos de clasificación que utiliza para sus datos cometan el menor número de fallos posibles.

La motivación por mejorar uno de estos algoritmos de clasificación ha inducido a su autor a desarrollar este **proyecto de investigación** centrado en el estudio, ejecución y análisis de una técnica que permite mejorar las prestaciones de un conjunto de clasificadores en problemas multiclase.

1.2. Entorno socio-económico

Durante los últimos años el uso de datos ha sufrido una crecida exponencial motivado por el uso de Internet (veáse Figura 1.1) y todos los aparatos que a la red se conectan. Según el Instituto Nacional de Estadística [5] "en el año 2016 en España, el 80,6 % de la población de 16 a 74 años ha utilizado Internet. [...] Conectarse a Internet es una práctica mayoritaria en los jóvenes de 16 a 24 años, con un 98,6 % en los hombres y un 98,2 % en las mujeres." El uso de las nuevas tecnologías es tan importante que el 4 de noviembre de 2016, con el inicio de la XII Legislatura de la democracia en España se creó el **Ministerio de Energía, Turismo y Agenda Digital** que se ocuparía de los asuntos referentes a las nuevas tecnologías planteando objetivos a largo plazo (Agenda Digital para España y para Europa 2020) y ocupándose de legislar aquellos aspectos de la sociedad que fuesen apareciendo en el futuro en relación a las TICs.

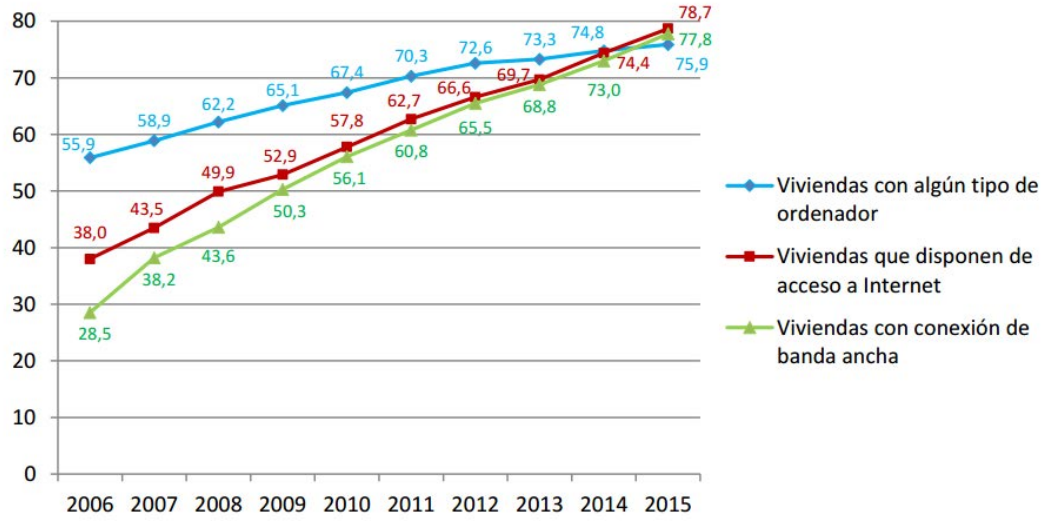


Figura 1.1: Evolución del equipamiento TIC en las viviendas
Fuente: INE. Datos de octubre de 2015.

Por otro lado está el crecimiento, también exponencial del **Internet de las cosas**, es decir, la interconexión digital de objetos cotidianos con Internet y viceversa. Estos objetos van desde luces, termostatos y puertas de una casa hasta coches inteligentes pasando por dispositivos incorporados en el cuerpo de las personas como marcapasos o cámaras y accesos de seguridad con inteligencia

artificial. Estos están produciendo una cantidad de datos por minuto del orden de miles de *Terabytes*, datos que pueden ser tratados con técnicas de *Big data* y aprendizaje automático.

Por último caben destacar los datos personales con los que las empresas identifican a sus clientes por medio de registros en portales webs y comercio electrónico o a través de tarjetas de fidelización en tiendas físicas. Con estos datos, es posible modelar hábitos de consumo para poder establecer estrategias de *marketing* más especializadas o crear productos más dirigidos al público objetivo.

La inversión en *Big data* y *machine learning* es cada vez más importante en grandes empresas como indica el informe de la Sociedad de la Información en España de Fundación Telefónica [6] y muestra la Figura 1.2 extraída de dicho informe.

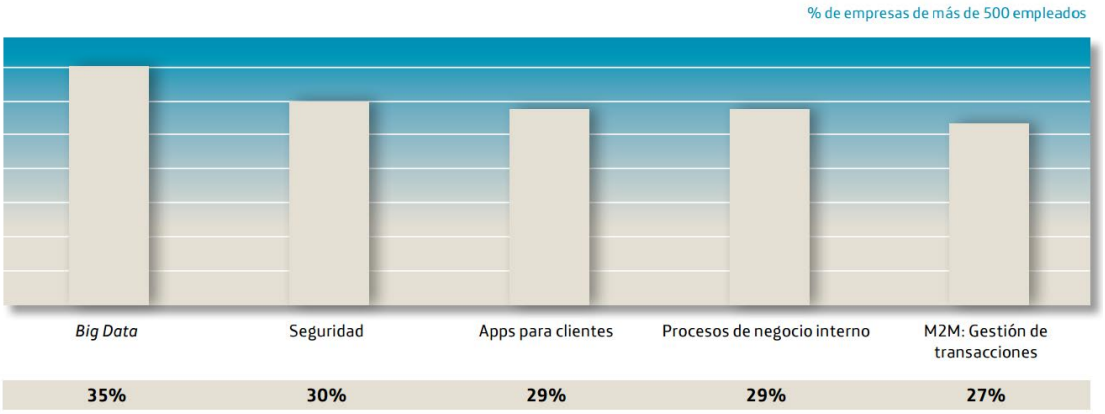


Figura 1.2: Tipo de soluciones sobre las que se han realizado inversiones digitales en 2016

Fuente: Telefónica. Datos de julio de 2016.

Centrado en las técnicas de *machine learning*, observando la Figura 1.3 que muestra un *Hype cicle*[1] o ciclo de sobreexpectación de la empresa ***Gartner***, se puede apreciar cómo los métodos de conjunto (en la gráfica *ensemble learning*), objeto de estudio del presente TFG, están en la zona de la **rampa de consolidación**, es decir, se empiezan a obtener beneficios para las empresas que utilizan dicha tecnología pero se siguen buscando segundas y terceras generaciones de esta.

[1]Representación gráfica de la madurez, adopción y aplicación comercial de tecnologías específicas, publicado de forma anual, que proporciona una perspectiva transversal de las tendencias de la industrias tecnológicas emergentes, ayudando a discernir si nos encontramos ante una sobreexpectación o ante una tecnología viable.

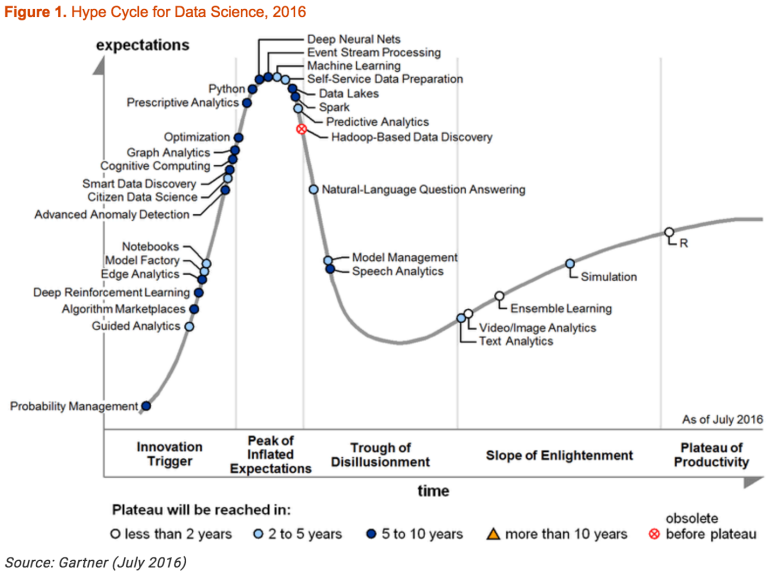


Figura 1.3: Ciclo de sobreexpectación Gartner 2016.
Fuente: Gartner. Datos de julio de 2016.

1.3. Marco regulatorio

El proyecto que ocupa el presente TFG no está sujeto a ningún marco regulatorio ya que las técnicas de aprendizaje automático no tienen marco legal establecido. Sin embargo, en lo relativo al tratamiento y uso de datos está en vigor en España la Ley Orgánica 15/1999, de 14 de diciembre de 1999, de Protección de Datos de Carácter Personal. (LOPD) [7]. Tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor, intimidad y privacidad personal y familiar.
A nivel europeo, el Reglamento europeo 2016/679 [8] aprobado el 27 de abril de 2016 será aplicable a partir del 25 de mayo de 2018 derogando la anterior directiva 95/46/CE (Reglamento general de protección de datos). Dicho Reglamento unifica y moderniza la normativa europea sobre protección de datos, permitiendo a los ciudadanos un mejor control de sus datos personales y a las empresas aprovechar al máximo las oportunidades de un mercado único digital, reduciendo la burocracia y beneficiándose de una mayor confianza de los consumidores.
Para las simulaciones del presente trabajo, se han utilizado bases de datos de carácter público obtenidas del repositorio de *machine learning* de la UCI [9]. Además, ninguna de ellas contiene datos de carácter personal.

1.4. Objetivos del trabajo

El objetivo principal de este trabajo es analizar, diseñar e implementar una técnica que permita mejorar las prestaciones de un conjunto de clasificadores base. Esta técnica, conocida comúnmente de su nombre en inglés, *class-switching*, se

basa en la introducción de diversidad en las etiquetas de las muestras de entrenamiento de cada clasificador base.

Para comprobar si se han obtenido resultados positivos en prestaciones, se comparará la tasa de error cometida por el conjunto de clasificadores, cuando se aplica dicha técnica y se utilizan diferentes métodos de combinación de las salidas de los mismos, con el error de base o *baseline*, que es el que obtendría con un único clasificador base.

Utilizar esta técnica en conjuntos de clasificadores no es algo novedoso. A este respecto, los cambios propuestos en este TFG con respecto a otras publicaciones que hacen uso del método *class-switching* son:

- Se pretende estudiar el algoritmo con dos tipos de clasificadores base diferentes a los utilizados previamente: clasificadores basados en perceptrones multicapa (MLPs) y clasificadores basados en el algoritmo k-NN.

- Se utilizarán dos técnicas diferentes de combinar las salidas obtenidas de los distintos clasificadores base para estudiar si existen mejoras de una con respecto a la otra.

- Se tendrán en cuenta bases de datos diferentes entre sí en cuanto al número de muestras y características, así como número de clases distintas y proporciones de estas que las componen.

1.5. Organización de la memoria

La memoria se ha estructurado en los siguientes capítulos:

- Capítulo 1. Motivación y objetivos. Se presentan los aspectos que han motivado la realización del proyecto, el entorno socio-económico en que se enmarca así como el marco regulador que aplica. Por último se presentan los objetivos que se pretenden alcanzar.

- Capítulo 2. Descripción del algoritmo. Estado del arte previo referente al algoritmo *class-switching* e introducción a este de forma teórica. Además se incluye una pequeña descripción de los algoritmos de clasificación MLP y k-NN.

- Capítulo 3. Diseño de la solución técnica. Explicación de forma general de cómo se ha implementado y evaluado el algoritmo utilizando la herramienta de *software* matemático MATLAB.

- Capítulo 4. Análisis de resultados. Se presentan los resultados obtenidos para todos los modelos propuestos y se extraen las conclusiones a partir de estos.

- Capítulo 5. Gestión del proyecto. Descripción de las fases de ejecución del proyecto y de su presupuesto.

- Capítulo 6. Conclusiones y líneas futuras. Se exponen las conclusiones extraídas de la investigación del proyecto y se proponen líneas futuras con las que seguir estudiando los métodos explorados.

Capítulo 2

Descripción del algoritmo

En este capítulo se introduce el estado del arte previo referente al algoritmo bajo estudio y posteriormente se introduce el propio algoritmo *class-switching* de manera teórica y los algoritmos de clasificación MLP y k-NN.

2.1. Estado del arte

Los algoritmos de clasificación supervisada utilizan muestras o vectores $\mathbf{x}$ categorizados con etiquetas y que representan la clase a la que pertenecen. Utilizan un conjunto de entrenamiento con muestras de las que se conoce tanto $\mathbf{x}$ como y para generar modelos que permitan decidir la clase a la que pertenecería una nueva muestra o dato $\mathbf{x}$ no conocido anteriormente por el algoritmo. En este proyecto, el conjunto de etiquetas de clase puede tener más de dos elementos. En este caso se dice que el problema de clasificación es multiclase.

Los métodos de conjuntos en aprendizaje máquina intentan inducir una colección de diversos clasificadores base o aprendices que son precisos y complementarios, de modo que, combinando las decisiones de diferentes máquinas se consigan mejores resultados para las muestras previamente no vistas [10]. En otras palabras, el objetivo es generar, a partir de un conjunto de datos de entrenamiento, una colección de diversos clasificadores cuyos errores están incorrelados. Los clasificadores de conjuntos construidos de esta manera, suelen representar mejoras considerables con respecto a un único clasificador. A continuación se listan referencias para distintos clasificadores base que pueden formar el conjunto: árboles de un solo nodo [11], árboles de decisión [12][13], redes neuronales [14][15][16], SVMs [17][18], etc. En la Figura 2.1 se muestra un esquema general de la arquitectura que siguen los algoritmos basados en métodos de conjuntos.

Para crear diversidad, los métodos de conjuntos introducen perturbaciones en alguna de las etapas de generación de las máquinas individuales. Esas perturbaciones o modificaciones pueden ser introducidas en cada uno de los clasificadores base o en el conjunto de entrenamiento y suele ser, generalmente, la entrada de algún tipo de aleatoriedad. Gracias a esto, es posible generar diversos clasificadores.

Centrando el estudio en los clasificadores que introducen aleatoriedad en el

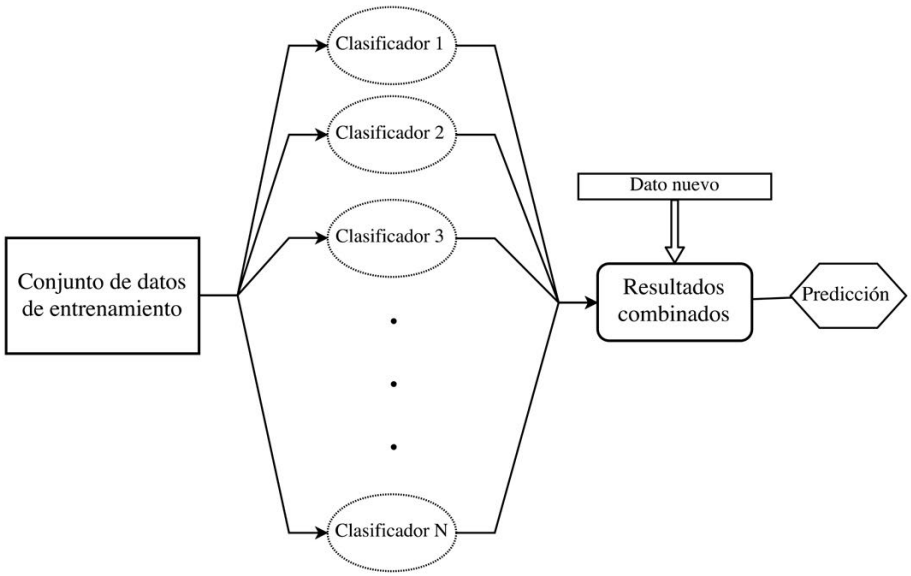


Figura 2.1: Esquema general de la arquitectura de un conjunto de clasificadores base.

conjunto de entrenamiento se encuentran técnicas como utilizar muestras *boostrap*[1], modificar la distribución o la importancia de algunas muestras o manipular algunas características o las etiquetas de los ejemplos de entrenamiento.

El método analizado en este trabajo corresponde al último de esos grupos de técnicas. Está basado en la modificación aleatoria de las clases de una parte del conjunto de muestras de entrenamiento para entrenar cada uno de los clasificadores base que componen el conjunto. Este tipo de algoritmos fueron introducidos por primera vez en [19] a través de la técnica *class-flipping*, más limitada que el *class-switching* [20]. En [20] se propone el método *class-switching* utilizando árboles de decisión sin podar generando buenos resultados para tasas de cambio de etiquetas de entrenamiento altas. En [21] se estudia el algoritmo *class-switching* para redes neuronales, las cuales aportan mejores resultados para tasas de cambio menores y con un menor número de clasificadores (≈ 200).

Teniendo en cuenta lo propuesto en [21], en este proyecto se estudia el método *class-switching* para clasificadores base basados en MLPs de una capa oculta y algoritmos k-NN sobre 5 bases de datos de diferentes número de características, clases, muestras y distribuciones.

[1] *Boostrap* es un algoritmo de aprendizaje automático diseñado para mejorar la estabilidad y precisión de métodos de aprendizaje máquina usados en clasificación y regresión. Reduce la varianza y ayuda a evitar el sobreajuste. Consiste en muestrar con reemplazo los datos de los que se dispone hasta completar un número de muestras predifinidas (o el total de las muestras). Es práctico cuando no se conocen las distribuciones de probabilidad a priori o son muy complejas, o cuando se dispone de pocas muestras.

2.2. El algoritmo *class-switching*

Aunque se puede utilizar para cualquier tipo de problema de aprendizaje supervisado, en este proyecto, el algoritmo *class-switching* ha sido utilizado para clasificación multiclase, esto es, utiliza un conjunto de muestras de entrenamiento $\mathbf{x}_{\text{train}}$ etiquetadas con nombres o valores nominales y_{train}.

Más detalladamente, se ubica en la familia de algoritmos de métodos de conjuntos, es decir, utiliza múltiples clasificadores sencillos para obtener mejores resultados que los que se obtienen con uno solo de ellos. Se caracteriza por introducir aleatoriedad en las etiquetas de entrenamiento y_{train}. Dado un clasificador base simple, se entrenan individualmente diferentes máquinas con lo propuesto en esta técnica. Todas estas máquinas tienen los mismos parámetros y arquitecturas pero se diferencian en las muestras con que son entrenadas. Estas muestras son las mismas y todas las disponibles para cada una de las máquinas pero con una proporción p de las etiquetas del conjunto de entrenamiento cambiadas aleatoriamente a una de otro valor de entre todas las que hay en la base de datos. Cada clasificador devuelve unas etiquetas estimadas de salida para muestras no conocidas pertenecientes al conjunto de test. Se combinan o agregan las salidas de todas las máquinas para obtener una salida única.

En este TFG se han tenido en cuenta dos formas de obtener la solución final a partir de la salida de todas las máquinas: el voto por mayoría de la salida dura y el promedio de las salidas blandas. A continuación, se explican brevemente ambas técnicas.

Voto por mayoría de la salida dura. Cada una de las máquinas base devuelve la etiqueta de la clase a la que pertenecerían las muestras no conocidas previamente. Como únicamente conoce el valor de la etiqueta $[1, 2, 3, 4..., N_{\text{clases}}]$ a esta forma de presentar el resultado se le conoce como **salida dura**. Teniendo las salidas de todas las máquinas para cada muestra, se decide, para el resultado final, aquella clase que más máquinas han decidido.

Promedio de las salidas blandas. Por otro lado, la mayoría de los algoritmos de *machine learning* ofrecen la posibilidad de obtener, para cada muestra no conocida previamente, un vector con las probabilidades de pertenecer a cada una de las clases, esta técnica es conocida como **salida blanda**. Con las probabilidades para cada muestra no conocida que devuelven todas las máquinas entrenadas es posible conocer el promedio de estas. Así, para cada muestra se selecciona aquella clase cuyo promedio sea mayor.

Cabe destacar la simplicidad de obtener una **salida dura** a partir de una **salida blanda** seleccionando aquella clase cuya $P(y|\mathbf{x})$ es mayor, donde P indica probabilidad.

2.2.1. Cambio de etiquetas

Como ya se ha comentado en el Capítulo 2, cambiar las etiquetas de las muestras de entrenamiento para generar clasificadores con perturbación a la entrada fue propuesto por primera vez en [19].

En este TFG el cambio de etiqueta del conjunto de entrenamiento que se realiza sobre un conjunto de proporción p fijo sigue el procedimiento propuesto en [21]. A continuación se describe dicho procedimiento.

El cambio de etiqueta aleatorio puede expresarse con:

$$\begin{aligned} P_{j\leftarrow i} &= \frac{p}{K-1} \quad \text{para} \quad j \neq i \\ P_{i\leftarrow i} &= 1-p \end{aligned} \tag{2.1}$$

donde $P_{j\leftarrow i}$ es la probabilidad de que una muestra cuya etiqueta original sea i pase a ser etiquetada como j. K es el número de clases de la base de datos, es decir, en los experimentos llevados a cabo en este TFG, $K = N_{\text{clases}}$.

Además, la proporción p de muestras cuya etiqueta ha sido cambiada, debe ser lo suficientemente baja para asegurar que, para todas las clases y regiones del espacio, hay una mayoría de muestras correctamente etiquetadas. Esta condición se cumple si $P_{j\leftarrow i} < P_{i\leftarrow i}$. Con esto y la Expresión (2.1) se obtiene:

$$p < \frac{(K-1)}{K} \tag{2.2}$$

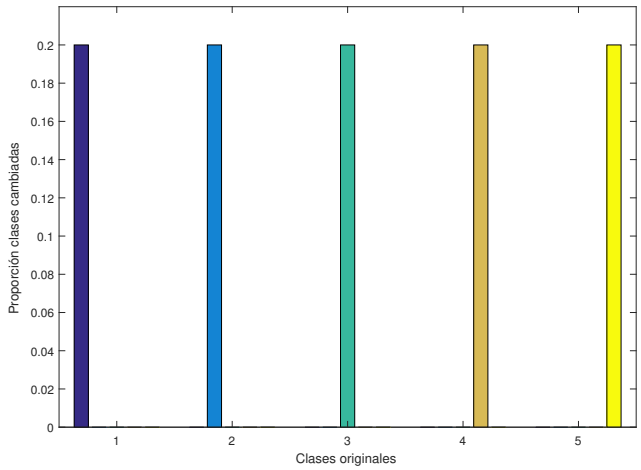
Para la Expresión (2.2), el ratio máximo de cambio está definido por:

$$\hat{p} = \frac{p}{p_{\max}} = \frac{pK}{(K-1)} \tag{2.3}$$

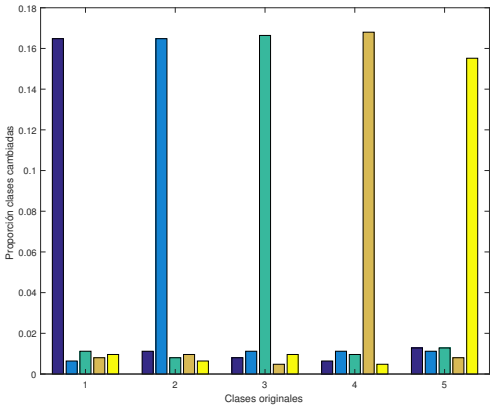
Esta variable $\hat{p}$, comprendida entre 0 y 1, será la que se variará para los experimentos llevados a cabo en este TFG en pasos de 0,2.

A modo de ejemplo, en la Figura 2.2 se muestra la distribución de las etiquetas cambiadas con respecto a las etiquetas originales para un ejmplo de base de datos compuesta por cinco clases con mismo número de muestras de cada clase, mientras que en la Figura 2.3 se muestra la distribución de clases en una base de datos que contiene tres clases y está desbalanceada. Esta es la principal diferencia de *class-switching* con respecto al algoritmo *class-flipping* propuesto en [19] el cual no se comporta bien para dichas bases de datos.

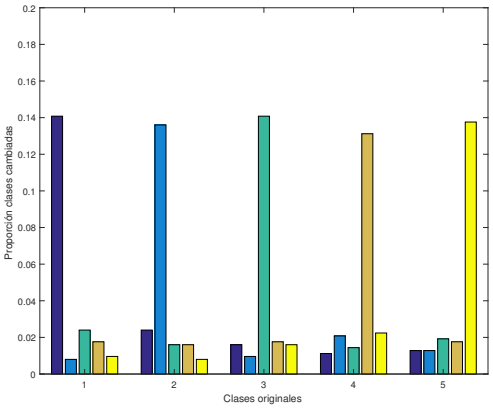
Se tienen en cuenta tasas de cambio $\hat{p} = 0, 1/5, 2/5, 3/5$ y $4/5$, que son los valores que han sido usados en las simulaciones de este proyecto y que se presentan en los resultados posteriores.



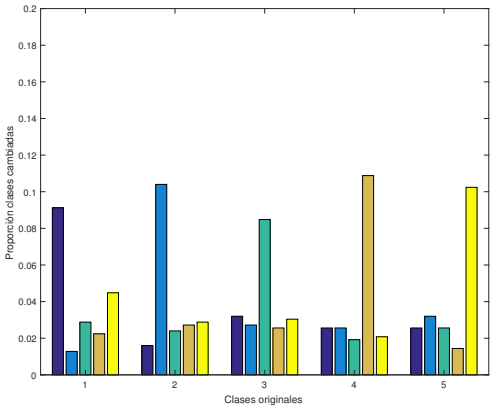
(a) $\hat{p} = 0$



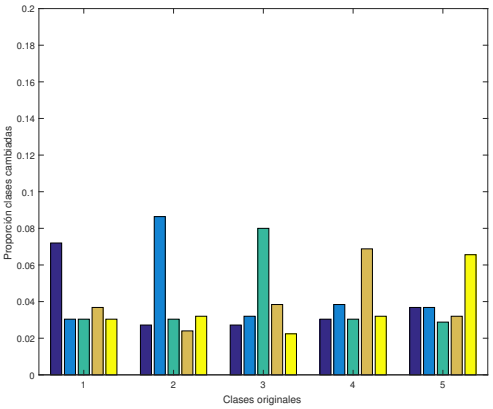
(b) $\hat{p} = 0{,}2$



(c) $\hat{p} = 0{,}4$

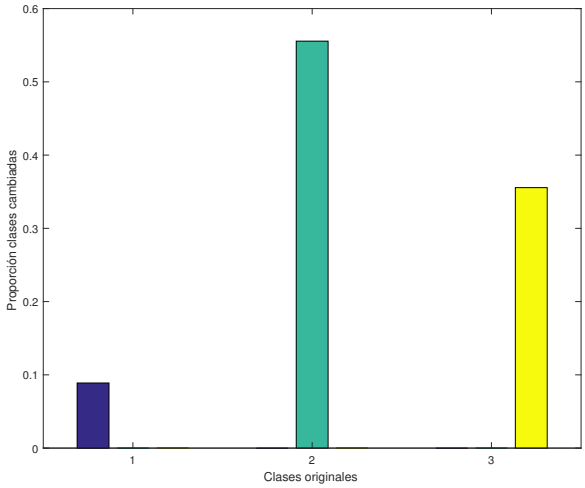


(d) $\hat{p} = 0{,}6$

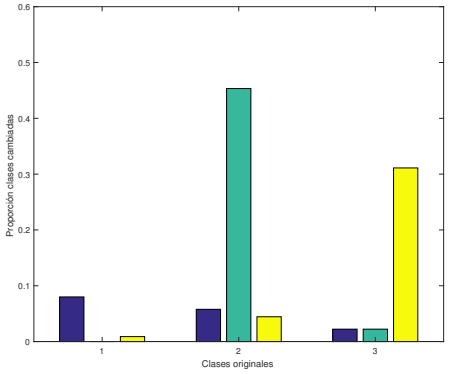


(e) $\hat{p} = 0{,}8$

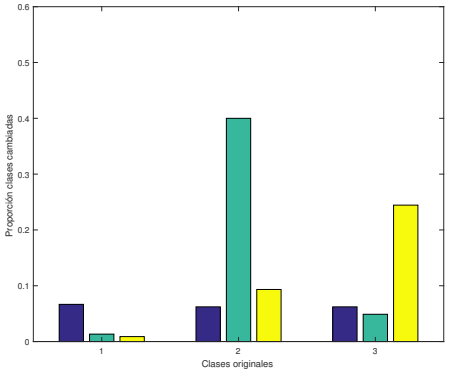
Figura 2.2: Distribución de las etiquetas según la tasa de cambio explorada, $\hat{p}$, para una base de datos balanceada.



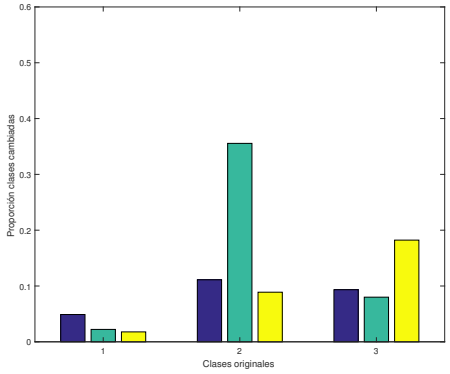
(a) $\hat{p} = 0$



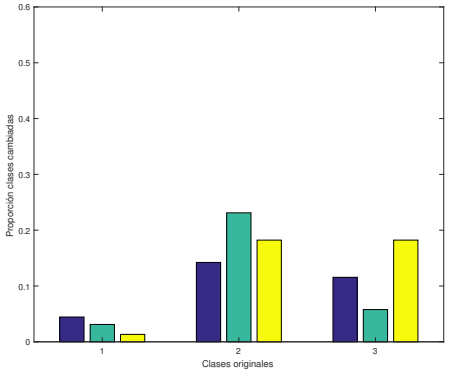
(b) $\hat{p} = 0{,}2$



(c) $\hat{p} = 0{,}4$



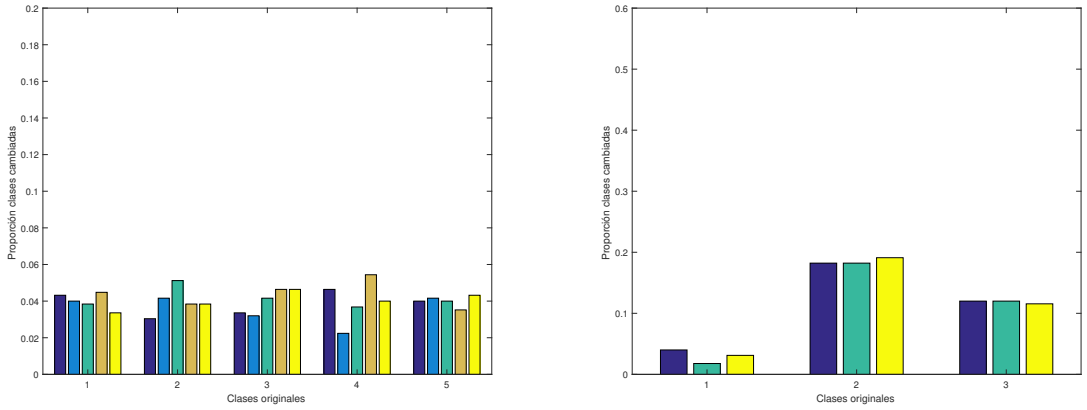
(d) $\hat{p} = 0{,}6$



(e) $\hat{p} = 0{,}8$

Figura 2.3: Distribución de las etiquetas según la tasa de cambio explorada, $\hat{p}$, para una base de datos desbalanceada.

En la Figura 2.4 se muestra un ejemplo de cómo quedarían las distribuciones para $\hat{p} = 1$. Se puede apreciar cómo en algunos casos, existen más muestras etiquetadas de una clase errónea. Esto hace que no tenga sentido explorar dicha tasa de cambio, por ello siempre se ha usado como mayor tasa de cambio $\hat{p} = 0{,}8$.



(a) $\hat{p} = 1$, base de datos balanceada (b) $\hat{p} = 1$, base de datos desbalanceada

Figura 2.4: Distribución de las etiquetas para $\hat{p} = 1$.

2.2.2. Clasificadores base

A continuación se introducen los dos clasificadores base utilizados para el estudio del algoritmo *class-switching*. Estos son dos de los algoritmos más simples en cuanto a dificultad técnica e implementación. No son los algoritmos con mejores prestaciones, de ahí que se espere que funcionen bien cuando se combinan y se utiliza el algoritmo explorado en este TFG.

Clasificadores MLPs

Una de las arquitecturas básicas utilizadas por las máquinas que componen los clasificadores estudiadas en este TFG son los perceptrones multicapa (MLPs) de una sola capa oculta, ya que se trata de máquinas inestables aunque presentan prestaciones aceptables. Los clasificadores se entrenan para minimizar el error cuadrático medio entre la salida deseada y la salida real del clasificador. Para ello, se inicializan aleatoriamente las componentes de los pesos del MLP con valores uniformemente distribuidos en el intervalo [-0.2, 0.2], y se realiza su actualización mediante el algoritmo de retropropagación. La tasa de aprendizaje α se ajusta para ser 0.01 (se ha comprobado experimentalmente que este valor es suficiente para asegurar la convergencia). Junto con el algoritmo de retropropagación, se emplea un algoritmo de detención prematura del entrenamiento tal y como se detallará en la Sección 3.1.1, utilizando el 80 % de los datos disponibles para entrenar el MLP y el 20 % restante para validar el diseño, es decir, para elegir los pesos correspondientes a la época que presenta un menor error sobre esta partición, reduciendo así los efectos perniciosos del sobreajuste.

Como se ha comentado, los MLPs explorados en este TFG consisten de una capa de entrada, una capa oculta y una capa de salida. En este tipo de arquitecturas, el número de neuronas en la capa de entrada (I), así como el número de neuronas en la capa de salida (O), están determinados por el tipo de problema a resolver. Concretamente, el número de neuronas en la capa de entrada coincide con el número de características o atributos extraídos de las muestras (es decir, es congruente con la dimensión de cada una de las bases de datos que se muestra en la Tabla 4.1), mientras que el número de neuronas de salida coincide con el número de clases a separar (nótese que en un problema de clasificación binaria, sería suficiente con tener una única neurona de salida). El número de neuronas en la capa oculta (H) depende de la complejidad del modelo. A modo de ejemplo, en la Figura 2.5 se puede observar la arquitectura de un MLP para muestras de cuatro características y con 3 posibles etiquetas de salida. La capa oculta tiene 5 unidades o neuronas. En los experimentos llevados a cabo en este TFG, según lo expuesto en [21] para todas las bases de datos se ha probado con $H = 3, 5, 7, 11, 15, 21$ y 27 con el objetivo de ahorrar en coste computacional. Además, se han evaluado perceptrones entrenados hasta $E = 100, 300, 500, 900$ y 1300 épocas de entrenamiento con el mismo objetivo y para evitar el sobreajuste que se cometería al intentar converger lo máximo. Como se explicará más adelante en la Sección 3.1.1 estos valores de H y E se estiman por validación cruzada.

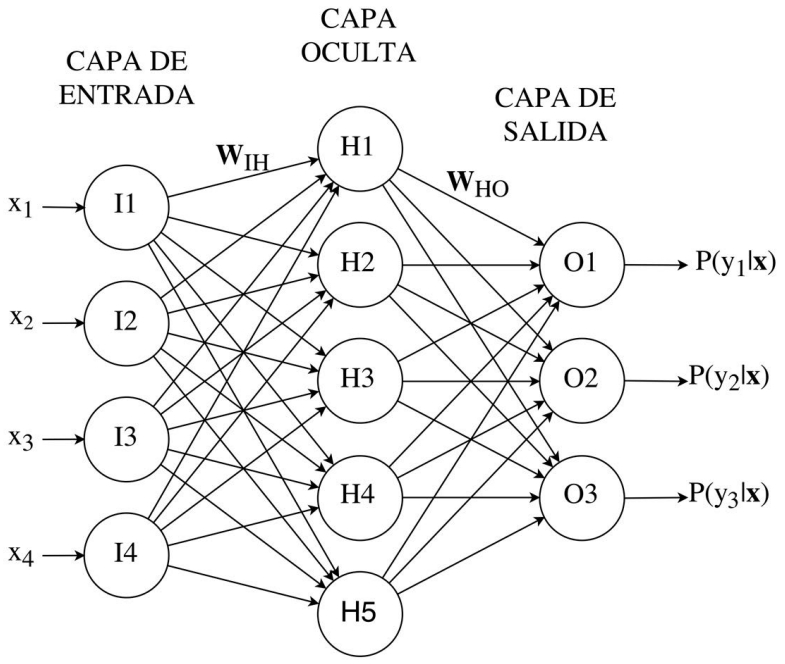


Figura 2.5: Ejemplo ilustrativo de la arquitectura de un MLP con $I = 4$ entradas, $O = 3$ salidas y $H = 5$ unidades en la capa oculta.

Cada una de las neuronas de salida $[O_1, O_2, O_3]$ devuelve la probabilidad a posteriori $P(y_i|\mathbf{x})$ de pertenecer a cada una de las clases $i = 1, 2, 3$ respectivamente. Cabe destacar cómo $\sum_{i=1}^{N_{\text{clases}}} P(y_i|\mathbf{x}) = 1$. A partir de estos valores es trivial conocer la clase clasificada como $\arg\max_i P(y_i|\mathbf{x})$.

Clasificadores basados en k-NN

El otro conjunto de clasificadores donde se ha aplicado la técnica de *class-switching* está basado en el algoritmo de clasificación k-NN (del inglés *k-nearest neighbours*). Es una de las aproximaciones más conocida dentro de las técnicas de clasificación supervisada y se basa en criterios de vecindad. La ventaja más inmediata que presentan las técnicas de clasificación por vecindad con respecto a otros métodos de clasificación, hace referencia a su simplicidad conceptual: la clasificación de un nuevo punto del espacio de representación se calcula en función de las clases, conocidas de antemano, de los puntos más próximos a él.

Tal y como se desprende de la afirmación anterior, la idea fundamental sobre la que se apoyan estas técnicas de clasificación, se basa en que las muestras pertenecientes a una misma clase, probablemente se encontrarán próximas en el espacio de representación.

En general, cualquier problema de clasificación abordado con un enfoque basado en criterios de vecindad se puede caracterizar del siguiente modo:

1. Se dispone de un conjunto de N_{train} muestras ya clasificadas llamado conjunto de entrenamiento.
2. Hay que clasificar una nueva muestra, $\mathbf{x}_i$, no perteneciente al conjunto de entrenamiento.
3. Existe una métrica entre los diferentes objetos del espacio de representación.
4. No se utiliza ninguna otra información acerca de la distribución de los parámetros estadísticos asociados al conjunto de entrenamiento.

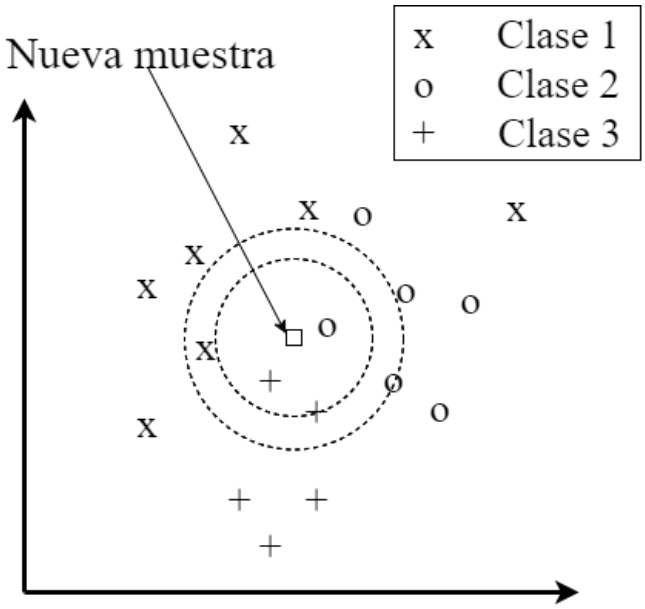


Figura 2.6: Representación gráfica del algoritmo k-NN cuando se utilizan $k = 3$ vecinos o $k = 5$ vecinos.

Por medio de este algoritmo, la clase asignada a una nueva muestra $\mathbf{x}_i$ será la clase más votada entre los k vecinos más próximos del conjunto de entrenamiento. En la Figura 2.6 se observa un ejemplo de la aplicación de este algoritmo para un problema de dos dimensiones o atributos con 3 clases 'x', 'o', '+'. En este caso si clasificásemos en función de los 3 vecinos más cercanos, nuestro candidato tiene próximos a él dos candidatos de la clase '+' y uno de la clase 'o', por tanto será clasificado con la clase '+'.

Sin embargo, cabe la posibilidad de que se produzca un empate en el número de votos de la clase más votada. En este caso, para seleccionar la clase a asignar, en este TFG se ha optado por seleccionar la clase de entre las que están empatadas que tiene el vecino más cercano. Cabe destacar cómo para un problema de dos clases, estos empates serían imposibles en caso de utilizar k número impar.

Si en la Figura 2.6 se tienen en cuenta las 5 muestras a menor distancia, es decir $k = 5$ se tendría un empate a 2 entre '+' y 'o' pero al haber una muestra 'o' más cerca del candidato, se decidirá a favor de esta clase. Véase cómo dependiendo del número k de vecinos escogido, se decide en favor de una clase u otra.

Cabe destacar que este algoritmo también ofrece **salida blanda**, es decir, probabilidades a posteriori de pertenecer a cada una de las clases. Esta se basa en la proporción de clases que hay entre los k vecinos mas cercanos. Así en la Figura 2.6, con $k = 3$ se tienen 1 muestra de la clase 'o' y dos de la clase '+' por lo que el clasificador devuelve un vector $y = \begin{bmatrix}0 & 0{,}33 & 0{,}66\end{bmatrix}$ donde la primera posición del vector es $P(y|\mathbf{x} =$'x'$)$, la segunda $P(y|\mathbf{x} =$'o'$)$ y la tercera $P(y|\mathbf{x} =$'+'$)$.
Para el caso de $k = 5$ se tiene $y = \begin{bmatrix}0{,}2 & 0{,}4 & 0{,}4\end{bmatrix}$ y es trabajo del diseñador deshacer los empates.

Capítulo 3

Diseño de la solución técnica

En este capítulo se detalla el funcionamiento del algoritmo y de cómo se ha desarrollado el código para su estudio. Como ya se ha introducido en el Capítulo 1, se han estudiado dos algoritmos base de clasificación sencillos para construir las máquinas: MLPs y k-NN. Primero se detalla el algoritmo para construir el primer tipo de máquinas, y a continuación se extrapola para el segundo.

Para extraer conclusiones se ha medido en todos los casos la tasa de error sobre el conjunto de test. Este conjunto está formado por muestras de las cuales se conoce su etiqueta de salida únicamente para comparar con la salida que aporta el algoritmo de aprendizaje automático tras ser entrenado y haber conocido las características de dichas muestras.

3.1. Implementación del algoritmo

Como ya se comentó en el Capítulo 2, para el estudio del algoritmo se han utilizado 5 bases de datos tomadas del repositorio UCI de *machine learning* [9]. Ya que estas bases de datos no distinguen entre muestras de entrenamiento y test, se ha decidido hacer un 5-*fold* para obtener 5 conjuntos de entrenamiento y test diferentes. Así, al promediar los resultados de cada ejecución se obtienen resultados más estables. Esta partición está representada en la parte superior de la Figura 3.2.

Cada par de conjuntos pasará por tres fases: selección del número de neuronas, cómputo del error de base y ejecución del algoritmo *class-switching*. Todas estas etapas pasan a detallarse a continuación.

3.1.1. Selección del número de neuronas y épocas de entrenamiento

En esta primera fase se busca, mediante validación cruzada o *cross-validation* (CV), el número de neuronas de la capa oculta óptimo (H) y épocas de entrenamiento (E) que se utilizarán en los clasificadores MLPs de los siguientes apartados. Como ya se ha comentado, esta selección es de gran importancia para evitar los efectos perniciosos del sobreajuste a los datos de entrenamiento.

Para obtener el valor de estos dos parámetros se ha utilizado CV de 5 subconjuntos de pares entrenamiento-validación con el objetivo de promediar los resultados y obtener una solución más significativa en términos estadísticos. Véase la parte inferior de la Figura 3.2 donde se muestra este procedimiento. Se tienen en cuenta las combinaciones de arquitecturas con $H = 3, 5, 7, 11, 15, 21$ y 27 unidades en la capa oculta entrenadas hasta $E = 100, 300, 500, 900$ y 1300 épocas.

Con los resultados de los 5 conjuntos de validación es posible obtener un promedio del error con respecto al número de neuronas y épocas. Este error representa el número medio de errores cometidos al etiquetar las muestras del conjunto de validación. Se selecciona el par H-E que mejores prestaciones proporcione, es decir, aquel cuyo promedio de errores cometidos para los cinco conjuntos de validación sea menor.

3.1.2. Cómputo del error de base

Con el número de neuronas (H) y épocas (E) obtenidas en el paso anterior se diseña un nuevo clasificador basado en un MLP. En este caso se entrena una única red con el conjunto de entrenamiento y se mide el error a través del conjunto de test.

Para paliar el efecto de la inicialización aleatoria en los pesos de las capas del clasificador se realizan 10 repeticiones del mismo experimento de forma independiente promediando la tasa de error. Este es el error que el algoritmo de *class-switching* intenta mejorar, por ello es conocido como **error de base**.

3.1.3. Ejecución del algoritmo class-switching

Como ya se ha explicado en el Capítulo 2 se han tenido en cuenta tasas de cambio de $\hat{p} = 0/5, 1/5, 2/5, 3/5$ y $4/5$. Para cada valor de $\hat{p}$ se entrenan 200 MLPs o máquinas a cuyas entradas se introducen, además de las características de las muestras de entrenamiento, las etiquetas de estas modificadas como se ha detallado en la Sección 2.2.1. Cada máquina es entrenada con el número de neuronas y épocas obtenidas en la primera fase descrita en la Sección 3.1.1. Se ha seleccionado 200 como el número de máquinas porque en ese punto, la mayoría de las simulaciones han convergido a sus mejores prestaciones.

Con las muestras de test, las máquinas devuelven como resultado dos matrices, ambas de dimensión $N_{\text{train}} \times N_{\text{clases}}$. Una de ellas se compone de las probabilidades a posteriori de pertenecer a la clase j para todas las muestras $\mathbf{x}_i$. La otra, para cada muestra $\mathbf{x}_i$ devuelve una fila de la matriz que indica, de forma binaria (veáse Expresión 3.1), la clase que se clasifica.

Como ya se ha comentado en la Sección 2.2 se utilizan ambos métodos para decidir la clase a la que pertenece la nueva muestra tal y como se detalla a continuación.

Voto por mayoría de la salida dura. En este caso, se tiene una matriz binaria con valores 0 y 1 que representan valores nominales como se puede ver en la Expresión 3.1. Con esta, se lleva a cabo una elección de "voto por mayoría" donde se clasifica la clase que más máquinas hayan decidido como salida.

Para evitar empates, en caso de haberlos, se desempata con una máquina entrenada sin modificación en las entradas del conjunto de entrenamiento. La tasa de error viene dada por el número medio de clasificaciones fallidas sobre el conjunto de test.

$$\begin{aligned} \textbf{Etiqueta nominal} &\leftrightarrow \textbf{Etiqueta binaria} \\ 1 &\leftrightarrow 10\cdots00 \\ 2 &\leftrightarrow 01\cdots00 \\ &\vdots \\ N_{\text{clases}}-1 &\leftrightarrow 00\cdots10 \\ N_{\text{clases}} &\leftrightarrow 00\cdots01 \end{aligned} \tag{3.1}$$

Promedio de las salidas blandas. Se tiene una salida blanda de clasificación con valores en el intervalo $[0, 1]$ para cada muestra de test $\mathbf{x}_i$. En este caso, se ha optado por calcular la media aritmética de estas para todas las salidas de todas las máquinas entrenadas. Para cada muestra se tienen N_{clases} probabilidades de pertenecer a cada una de las clases, $P(y_i|\mathbf{x})$. El algoritmo clasificará las muestras atendiendo a la probabilidad promedio más alta. Como en el caso anterior, tras esta clasificación se computa el error como el número medio de muestras de test mal categorizadas.

Nótese cómo, a pesar de la relación entre una y otras salidas, como se detalló en la Sección 2.2, es frecuente que no siempre se obtengan los mismos resultados al combinar las salidas de todos los clasificadores. En la Figura 3.1 se muestra un ejemplo sencillo de cómo, para la misma muestra de test, se puede obtener un resultado diferente al aplicar el método de voto por mayoría de la salida dura o promedio de las salidas blandas.

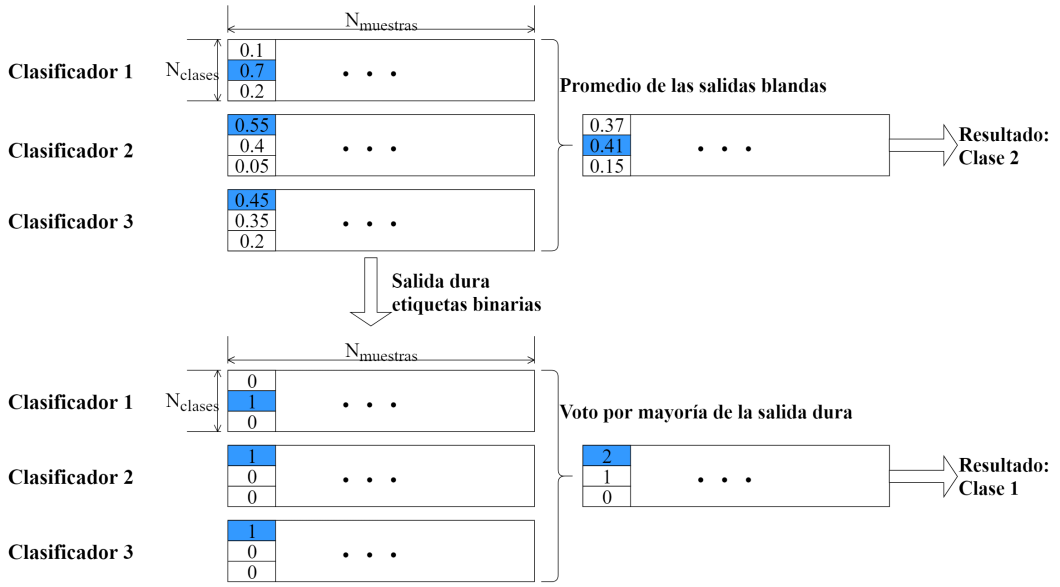


Figura 3.1: Ejemplo de la diferencia entre resultados que puede ofrecer la salida dura con respecto a la salida blanda

3.1.4. Algoritmo con k-NN

Se ha utilizado la distancia euclídea[1] para clasificar las muestras ya que el objetivo es que las máquinas que componen el conjunto de clasificadores sean lo más sencillos posibles.

En este caso, el único parámetro a seleccionar en la primera fase es el número de vecinos, k, a tener en cuenta para la clasificación. Como en el caso del número de neuronas y épocas para los MLPs, este valor se estima mediante validación cruzada con el conjunto de entrenamiento generando 5 subconjuntos de validación de igual manera que para el otro tipo de clasificadores. Se han tenido en cuenta valores de k de 1 a 50 aunque se ha comprobado experimentalmente cómo, para las bases de datos bajo estudio, siempre eran seleccionados un número de vecinos menores a 10.

Con este número de vecinos y los conjuntos de entrenamiento y test se calcula el error de base. En este caso, al no introducir aleatoriedad en la inicialización del clasificador (al contrario que en los pesos de los MLPs), no es necesario repetir el experimento para obtener un valor medio. Esto se debe al carácter determinista del algoritmo.

Por último se ejecuta el algoritmo *class-switching* para $\hat{p} = 0/5, 1/5, 2/5, 3/5, 4/5$ y se obtiene, para cada muestra del conjunto de test, las dos salidas antes mencionadas: la **salida blanda** que indica un nivel de confianza de pertenecer a cada una de las clases según el número de vecinos pertenecientes a cada clase de entre

[1]La distancia euclídea entre dos vectores P y Q se define como $D_E(P,Q) = \sqrt{\sum_{i=1}^{N}(p_i - q_i)^2}$ donde N es el número de dimensiones o características de las muestras, P es el candidato a etiquetar y Q es una muestra cualquiera del conjunto de entrenamiento.

los k más cercanos y la **salida dura** que únicamente da a conocer la clase a la que se clasificaría cada muestra.

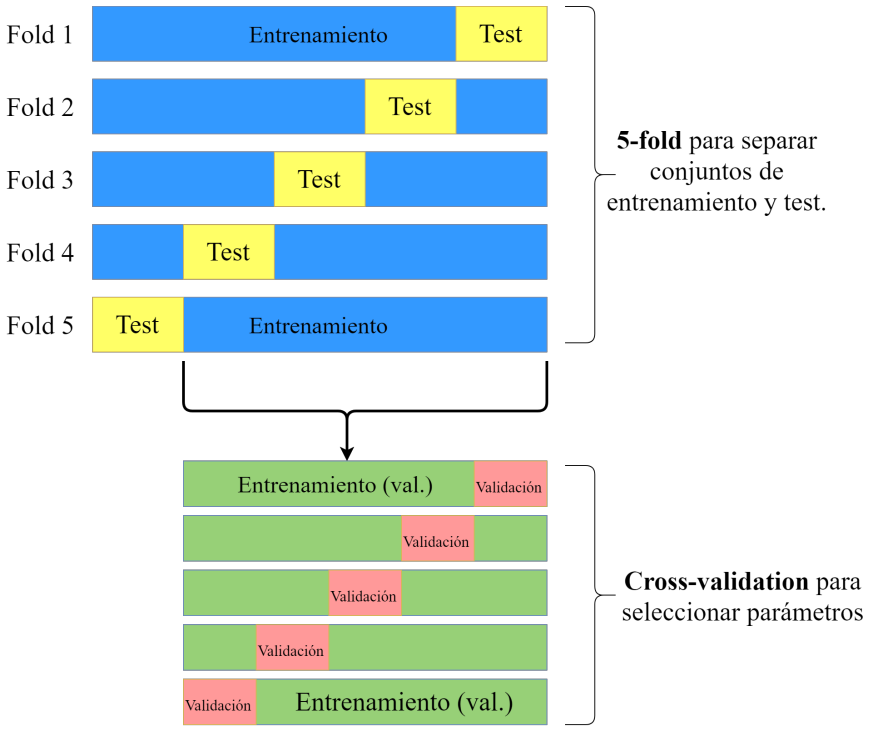


Figura 3.2: Representación de la doble división de las bases de datos para 5-fold y validación cruzada

3.2. Librerías MATLAB

Como ya se ha indicado, para la implementación de las simulaciones se ha utilizado la herramienta de *software* matemático MATLAB. Se ha considerado el uso de dos *toolboxes* para implementar los algorimos MLP y k-NN ya que el objetivo del TFG no es la implementación de estos, sino el estudio de las prestaciones del algoritmo *class-switching*, y MATLAB ofrece estos algoritmos optimizados. A continuación se describen las dos *toolboxes* empleadas.

***Statistics and Machine Learning Toolbox* [22]:** Ofrece funciones para describir, analizar, modelar datos y representar los resultados. La herramienta incluye algoritmos de *machine learning* de aprendizaje supervisado y no supervisado como árboles de decisión, k-NN, k-means y algoritmos de *clustering*. Se ha utilizado para los experimentos con máquinas de tipo k-NN. De esta herramienta se han utilizado las funciones `fitcknn` y `predict` para entrenar cada máquina con las muestras de entrenamiento y clasificar las muestras de test, respectivamente.

***Neural Network Toolbox* [23]:** Esta herramienta proporciona algoritmos, modelos y aplicaciones para crear, entrenar, visualizar y simular redes neuronales,

entre ellas el MLP. Se han utilizado las funciones `patternnet` y `train` para crear la red y entrenarla respectivamente.

Capítulo 4

Resultados experimentales

En este capítulo se introducen en primer lugar, las características de las bases de datos utilizadas para posteriormente mostrar los resultados obtenidos en los experimentos que se han llevado a cabo con dichas bases de datos, para los dos algoritmos de clasificación base utilizados en los experimentos.

4.1. Bases de datos

En la Tabla 4.1 se muestran las características principales de las bases de datos utilizadas para estudiar el algoritmo. Como ya se ha comentado anteriormente, estas están compuestas por un solo conjunto de muestras que ha sido dividido en conjuntos de entrenamiento y test mediante un 5-*fold*.

En cada uno de los problemas de clasificación, el conjunto de datos ha sido normalizado, es decir, la media muestral de los valores normalizados es nula y su varianza muestral unitaria.

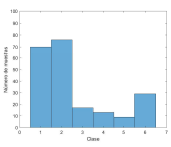
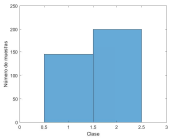
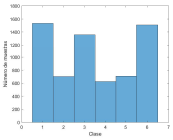
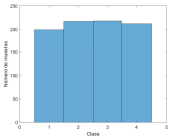
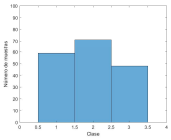
Base de datos	Nº de Muestras	Nº de Clases	Nº de Atributos	Distribución
Glass	214	7	9	
Liver	345	2	6	
Satimage	6435	6	36	
Vehicle	846	4	18	
Wine	178	3	13	

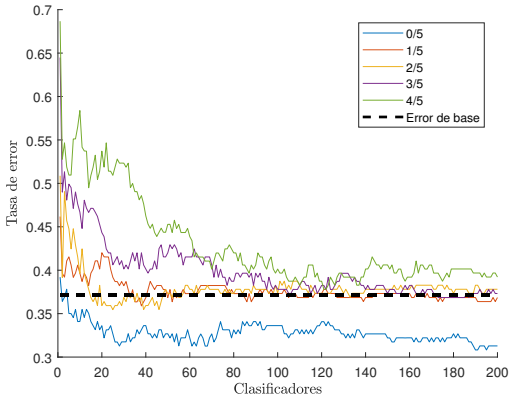
Tabla 4.1: Bases de datos utilizadas y sus características principales.

4.2. Resultados

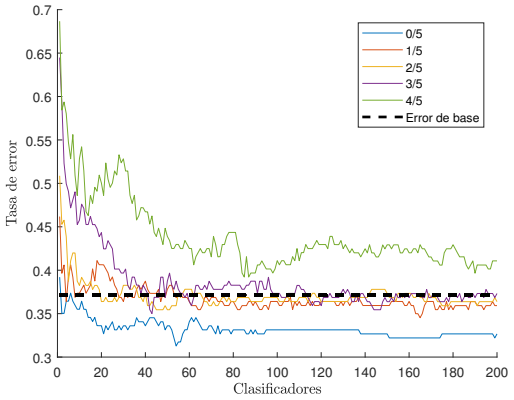
En esta sección se presentan los resultados obtenidos para las simulaciones con cada una de las bases de datos utilizando el algoritmo *class-switching*, tanto cuando el conjunto de clasificadores utiliza clasificadores basados en MLPs como cuando utiliza algoritmos de clasificación basados en el k-NN.

Las Figuras 4.1, 4.2, 4.3, 4.4 y 4.5 muestran el error de base en línea discontinua, así como los resultados obtenidos para distintos valores de $\hat{p}$ como función del número de máquinas utilizadas en el conjunto de clasificadores, para las bases de datos **glass**, **liver**, **satimage**, **vehicle** y **wine**, respectivamente.

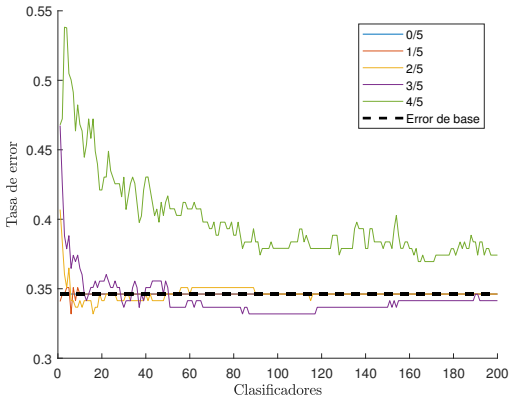
Nótese que al pie de cada figura, se indica el nombre de la base de datos, el tipo de clasificador base utilizado así como la técnica utilizada para combinar las salidas de los clasificadores base.



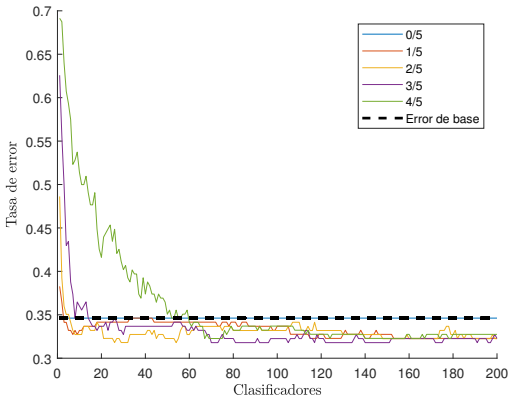
(a) Glass, MLP, máquinas de salida dura



(b) Glass, MLP, máquinas de salida blanda

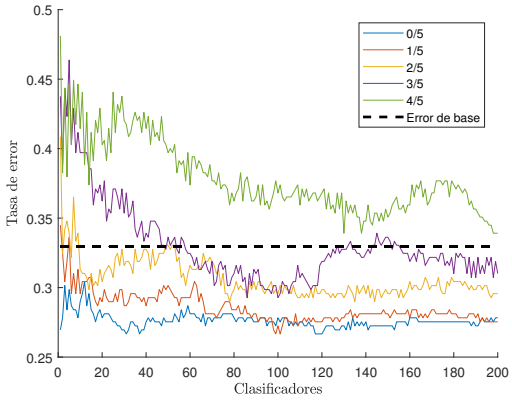


(c) Glass, k-NN, máquinas de salida dura

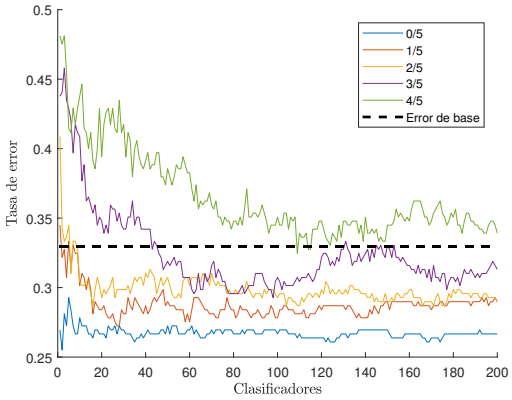


(d) Glass, k-NN, máquinas de salida blanda

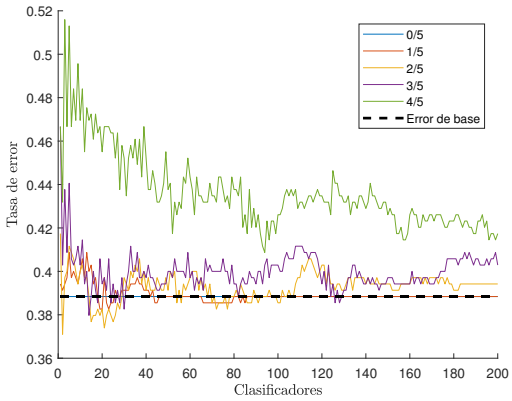
Figura 4.1: Resultados de los experimentos sobre la base de datos **glass**.



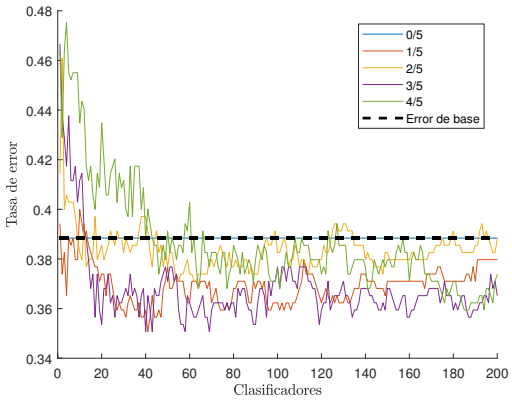
(a) Liver, MLP, máquinas de salida dura



(b) Liver, MLP, máquinas de salida blanda

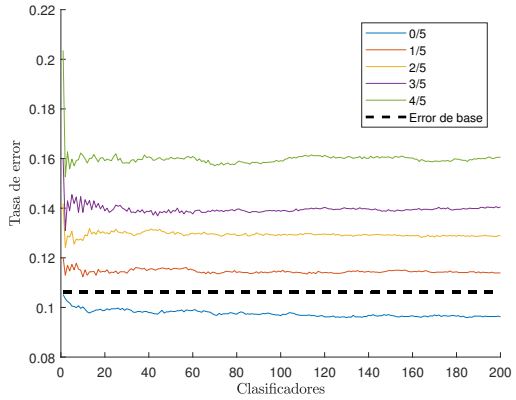


(c) Liver, k-NN, máquinas de salida dura

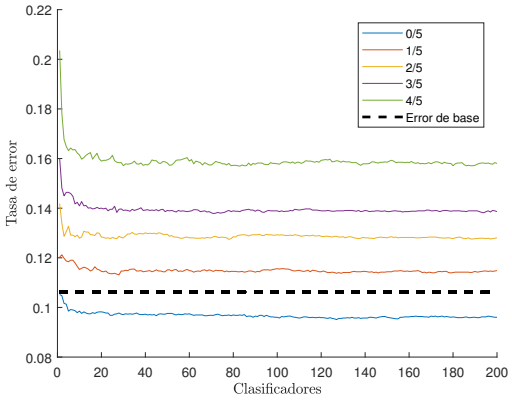


(d) Liver, k-NN, máquinas de salida blanda

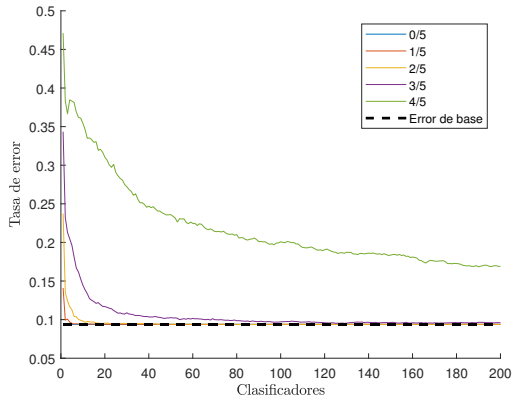
Figura 4.2: Resultados de los experimentos sobre la base de datos **liver**.



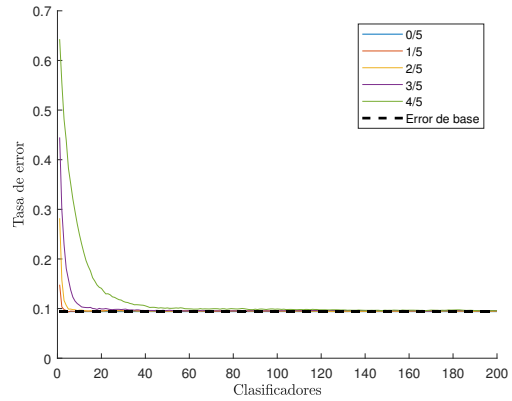
(a) Satimage, MLP, máquinas de salida dura



(b) Satimage, MLP, máquinas de salida blanda

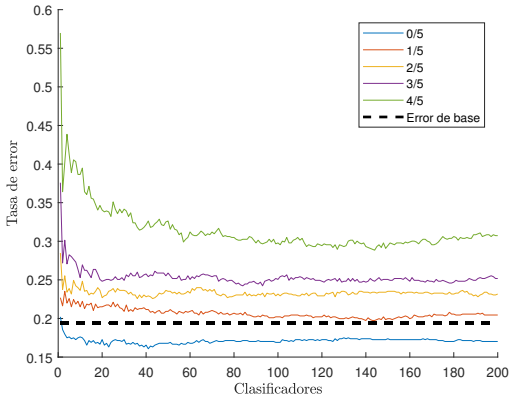


(c) Satimage, k-NN, máquinas de salida dura

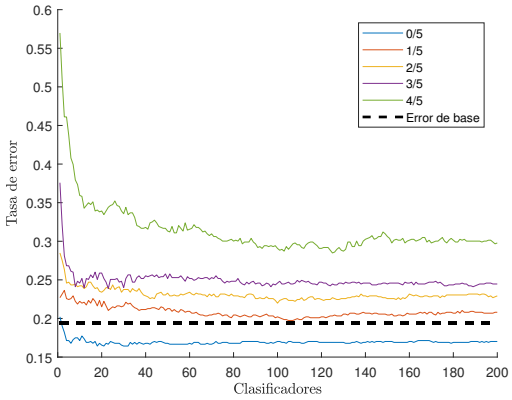


(d) Satimage, k-NN, máquinas de salida blanda

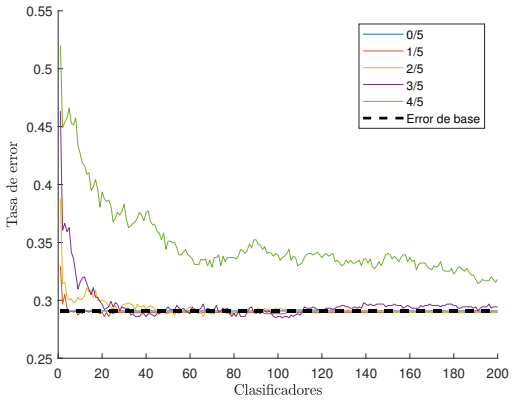
Figura 4.3: Resultados de los experimentos sobre la base de datos **satimage**.



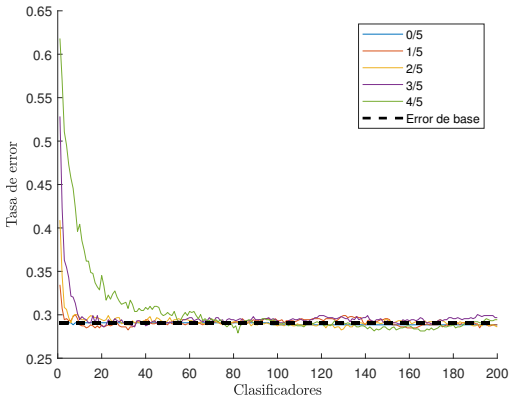
(a) Vehicle, MLP, máquinas de salida dura



(b) Vehicle, MLP, máquinas de salida blanda

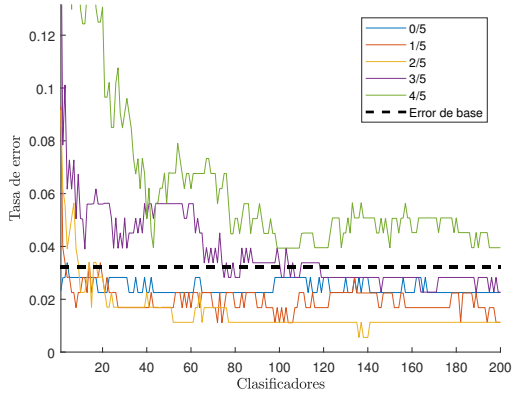


(c) Vehicle, k-NN, máquinas de salida dura

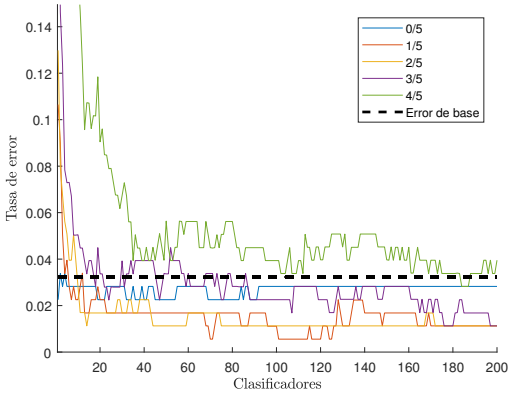


(d) Vehicle, k-NN, máquinas de salida blanda

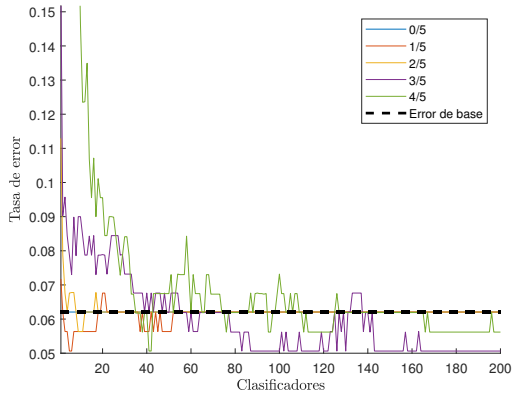
Figura 4.4: Resultados de los experimentos sobre la base de datos **vehicle**.



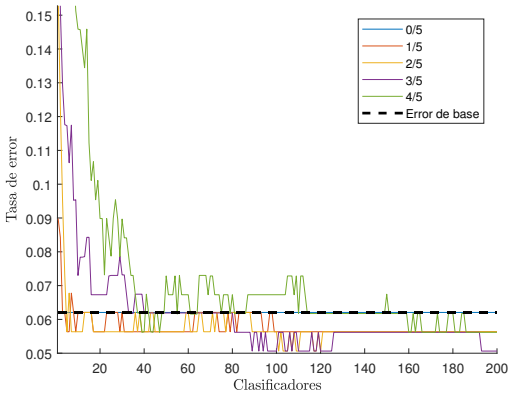
(a) Wine, MLP, máquinas de salida dura



(b) Wine, MLP, máquinas de salida blanda



(c) Wine, k-NN, máquinas de salida dura



(d) Wine, k-NN, máquinas de salida blanda

Figura 4.5: Resultados de los experimentos sobre la base de datos **wine**.

4.3. Análisis de resultados

Para analizar los resultados, se analiza la **tasa de error de test** cometida por el conjunto de los clasificadores, esto es, la proporción de muestras del conjunto de test mal clasificadas por el conjunto de clasificadores con respecto a las etiquetas originales de dichas muestras. Obviamente, tasas menores implican mejores prestaciones y viceversa.

A primera vista, como era de esperar, se observa que el error de base de un clasificador basado en un MLP es menor que el que se obtiene cuando el clasificador hace uso del algoritmo k-NN. En este respecto, puede observarse que en la mayoría de los casos, la técnica de *class-switching* consigue reducir más el error cuando se utilizan clasificadores basados en MLPs que cuando se hace uso del algoritmo k-NN. Esto es debido a que el algoritmo de clasificación k-NN es más robusto ante los ruidos de datos.

Por otro lado, puede verse cómo tasas $\hat{p}$ de cambio bajas ($\hat{p}$=1/5 o 2/5) suelen dar mejores resultados, aunque en la mayoría de las bases de datos, para clasificadores basados en MLPs, los mejores resultados se consiguen con una tasa de cambio de 0, es decir, únicamente combinando clasificadores entrenados con el conjunto de entrenamiento sin modificar. Como es lógico, con 1 máquina, el error cometido es muy similar al error de base y a partir de la máquina 2 empieza a decrementar. Cabe destacar que en el caso del algoritmo k-NN, para esta tasa de cambio ($\hat{p} = 0$), el error coincide con el error de base. Esto es debido al carácter determinista del algoritmo.

Examinando las Figuras 4.3 y 4.4, se concluye que, para bases de datos con mayor número de muestras (**satimage** y **vehicle**), la tasa de error converge tras entrenar y clasificar con relativamente pocas máquinas.

Diferenciando entre las dos técnicas de combinación de resultados (voto por mayoría de la salida dura y promedio de las salidas blandas), se aprecian diferencias significativas en favor del método de salidas blandas que aporta menor variabilidad en la tasa de error a medida que aumenta el número de clasificadores que forman el conjunto de máquinas. Además para bases de datos con menor número de muestras y de clases, se obtienen resultados ligeramente mejores con este método.

Por último, cabe destacar que el tiempo de computación necesario para ejecutar el algoritmo *class-switching* es del orden de $N_{\text{clasificadores}}$ veces el tiempo de ejecución de un clasificador base.

Como conclusión general, se obtiene que el algoritmo *class-switching* funciona mejor cuanto menor sea el número de muestras de la base de datos y menor el número de clases diferentes que la componen. Asimismo, siempre será mejor utilizar el método nombrado como **promedio de las salidas blandas** para combinar los resultados de cada clasificador base.

Capítulo 5

Gestión del proyecto

La norma UNE-ISO 21500:2013 [24] recoge los procedimientos básicos necesarios para llevar a cabo la gestión y dirección de proyectos. Un proyecto es un conjunto único de procesos que consta de actividades coordinadas y controladas, con fechas de inicio y fin, que se llevan a cabo para lograr los objetivos de un proyecto.

En este capítulo se definen la planificación del proyecto y de todas sus fases con sus fechas de inicio y fin y los costes del material y los recursos humanos que constituyen el presupuesto del proyecto.

5.1. Planificación

A continuación se definen las fases del proyecto y sus fechas de inicio y fin. En la Tabla 5.1 se detallan las horas reales dedicadas a cada una de estas. La Figura 5.1 presenta de manera visual las etapas a través del tiempo.

- Búsqueda de tutor y tema del proyecto *(10 de enero del 2017 a 25 de enero del 2017)*: búsqueda de tutor orientada al tema de *machine learning* dentro de los ofrecidos en el tablón de proyectos de la Universidad Carlos III de Madrid y del Departamento de Teoría de la Señal y Comunicaciones.
- Planteamiento inicial del problema *(30 de enero de 2016)*: estudio preliminar del tema planteado con la tutora, definición de requisitos y objetivos a cumplir de forma superficial.
- Documentación y estudio del problema *(6 de febrero de 2017 a 15 de marzo de 2017)*: recopilación, lectura y comprensión de artículos relacionados con el proyecto.
- Diseño del algoritmo *(20 de marzo de 2017 a 12 de mayo de 2017)*: diseño, pruebas, corrección de errores del código que ejecuta el algoritmo en la herramienta MATLAB.
- Implementación de las simulaciones *(16 de mayo de 2017 a 26 de mayo de 2017)*: simulaciones en las distintas bases de datos, clasificadores base y métodos de obtener la solución a partir de las salidas de estos.

- Análisis de resultados *(18 de mayo de 2017 a 28 de mayo de 2017)*: estudio de los resultados obtenidos en las simulaciones.
- Elaboración de la memoria *(18 de mayo de 2017 a 21 de junio de 2017)*: estructuración y redacción de la memoria del proyecto.
- Defensa del proyecto *(4 de julio de 2017 a 10 de julio de 2017)*: preparación y exposición del trabajo realizado ante un tribunal de evaluación.

id.	**Tarea**	**Horas dedicadas**
α	Búsqueda de tutor y tema	4
β	Planteamiento inicial	2
γ	Documentación y estudio del problema	50
δ	Diseño del algoritmo	120
ϵ	Implementación de las simulaciones	22
ζ	Análisis de resultados	4
η	Elaboración de la memoria	80
θ	Defensa del proyecto	8
	Total	**290**

Tabla 5.1: Tiempo real, en horas, consumido en las etapas del proyecto

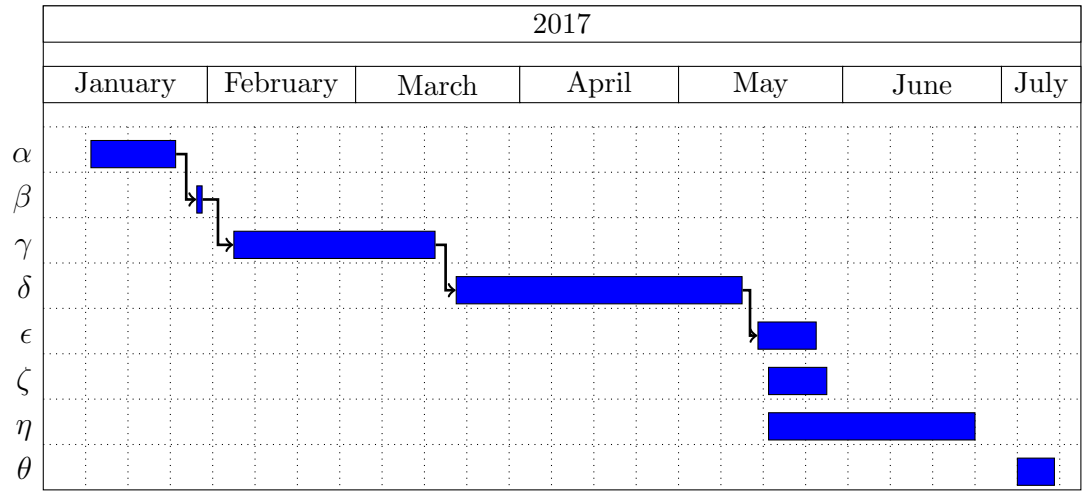


Figura 5.1: Diagrama de Gantt con las fases del proyecto y las dependencias entre ellas

5.2. Presupuesto

Los costes asociados al proyecto se pueden agrupar en los sigientes tipos

- Costes de recursos humanos.

- Costes de material *hardware* y licencias *software*.
- Costes indirectos.

5.2.1. Costes de recursos humanos

Los costes de personal se calculan a partir de las personas involucradas en el proyecto, las horas dedicadas y el coste por hora según su grado de responsabilidad dentro de éste. En este trabajo únicamente han participado el autor, como ingeniero y la tutora del Trabajo Fin de Grado, como jefa de proyecto. Los costes aparecen detallados en la Tabla 5.2.

Nombre	**Puesto**	**€/Hora**	**Horas totales**	**Coste total (€)**
Lorena Álvarez Pérez	Jefa de proyecto	40	25	**1000**
Adrián Vázquez Romero	Ingeniero	20	290	**5800**
			Total	**6800**

Tabla 5.2: Costes de recursos humanos implicados en el proyecto

5.2.2. Costes de material

En este proyecto no ha sido utilizado material *hardware* más allá de un ordenador donde trabajar con los programas *software*. En la Tabla 5.3 se detallan los costes de éstos.

Producto	**Precio (€)**	**Periodo de amortización (meses)**	**Tiempo utilizado (meses)**	**Coste total (€)**
Lenovo G510	800	60	6	**80**
Licencia Matlab Student	35	12	6	**17.5**
Statistics and Machine Learning Toolbox	20	12	6	**10**
Neural Network Toolbox	20	12	6	**10**
			Total	**117.5**

Tabla 5.3: Costes de los materiales usados para el proyecto

5.2.3. Costes indirectos

Los costes indirectos vienen dados por un porcentaje sobre la suma de los anteriores. Este porcentaje puede tomar diferentes valores según el ámbito del proyecto. Uno de los valores más comunes para trabajos realizados en la Universidad Carlos III de Madrid es el 15 %. Así se tiene que el coste indirecto en este proyecto es de 1037.63€.

5.2.4. Coste total

La Tabla 5.4 muestra los costes totales que supone el proyecto.

Concepto	**Coste (€)**
Costes personal	**6800**
Costes materiales	**117.50**
Costes indirectos	**1037.63**
Total	**7955.13**

Tabla 5.4: Costes totales

El coste total del proyecto son *siete mil novecientos cincuenta y cinco euros con trece céntimos*.

Capítulo 6

Conclusiones y líneas futuras

Este Trabajo Fin de Grado ha tenido como objetivo principal evaluar las prestaciones de una técnica, comúnmente utilizada en conjuntos de clasificadores, conocida como *class-switching* o cambio de etiqueta, en español. Inicialmente, se ha llevado a cabo desde un punto de vista teórico para posteriormente, mediante simulaciones, trasladar estos conceptos teóricos a situaciones prácticas.

Los dos ejes principales de este estudio han sido:

- Por un lado, analizar y comparar las prestaciones de utilizar clasificadores base que utilicen métodos de clasificación no estudiados hasta el momento cuando se aplica la técnica de *class-switching*, como son, clasificadores basados en perceptrones multicapa, así como clasificadores basados en el popular algoritmo de vecindad denominado k-NN.

- Por otra parte, se han analizado diferente esquemas de agregación o combinación de las salidas de los diversos clasificadores. Se han utilizado esquemas de "voto por mayoría" y esquemas de "promedio de las salidas blandas", los cuales permiten obtener resultados de clasificación más eficientes.

Para la realización de los experimentos, se han utilizado cinco bases de datos, comúnmente utilizadas en este tipo de problemas, con diferente número de características, número de clases y número de muestras. Todas las simulaciones han sido realizadas utilizando la herramienta de *software* matemático MATLAB.

Las conclusiones extraídas de este trabajo son las siguientes:

- Se ha comprobado experimentalmente que cuando el número de muestras del problema de clasificación, así como el número de clases que lo componen, son relativamente pequeños, el método de *class-switching* ofrece buenas prestaciones tanto para conjuntos de clasificadores base basados en MLPs como en algoritmos de clasificación k-NN, mejorando el error de base en la mayoría de los casos. Para bases de datos con un mayor número de muestras, la tasa de error converge rápidamente con relativamente pocas máquinas en el conjunto.

- Asimismo, se ha verificado gráficamente que las "mejores" tasas de cambio $\hat{p}$ son aquellas que suponen un cambio aleatorio del 20-40 % de las etiquetas de entrenamiento, aunque no siempre sucede esto ya que, en algunos casos, se ha comprobado que los mejores resultados se obtienen con una tasa de cambio de $\hat{p}$=0, es decir, únicamente combinando clasificadores entrenados con el conjunto de muestras de entrenamiento original.

Posteriormente se han comparado las prestaciones de los conjuntos de clasificadores en un escenario en el que las salidas se combinan utilizando el "voto por mayoría" y en el que, se agregan utilizando el esquema de "promedio de las salidas blandas". Se demuestra que las prestaciones son peores si no se aplica el esquema de "promedio de las salidas blandas" en la mayoría de los casos. Estas diferencias de prestaciones son notables cuando se utilizan clasificadores basados en el algoritmo k-NN y/o tasas de cambio grandes.

Finalmente se ha comentado el compromiso entre número de clasificadores que componen el conjunto y coste computacional. Obviamente, el coste computacional es mayor cuanto mayor es el número de máquinas que componen el conjunto de clasificadores y en términos de memoria, si se utilizan clasificadores basados en el algoritmo de clasificación k-NN, a mayor número de máquinas, mayor requerimiento de memoria.

Entre las líneas futuras de investigación que surgen tras la realización de este trabajo destacan:

- Estudio de otro tipos de clasificadores base como pueden ser, máquinas de vectores soporte o clasificadores de regresión logística, que en muchos problemas de clasificación permiten frecuentemente alcanzar mejores prestaciones con baja carga computacional en operación.

- Exploración de otras técnicas de introducción de diversidad en conjuntos de clasificadores como son *Bagging*, *Boosting*, etc.

- Aplicación una alternativa sencilla, denominada pre-énfasis, consistente en ponderar las muestras de entrenamiento de acuerdo con una función de énfasis que tiene en cuenta el carácter crítico de las mismas, es decir, su proximidad a la frontera de clasificación y el error de la mismas.

Bibliografía

[1] Facebook, "Applying machine learning science to Facebook products," https://research.fb.com/category/applied-machine-learning/, 2016, [Online; accessed 13-6-2017].

[2] S. Borger, "The city will help you live in it," http://www.research.ibm.com/cognitive-computing/machine-learning-applications/smart-cities.shtml#fbid=7T7qkVSa8zi, 2016.

[3] D. Stavens, S. Thrun, F. Li, A. Ng, S. U. C. S. Department, and S. U. D. of Computer Science, *Learning to Drive: Perception for Autonomous Cars*. Stanford University, 2011. [Online]. Available: https://books.google.es/books?id=uu9lgaQUF14C

[4] P. Felgaer, "Optimización de redes bayesianas basado en técnicas de aprendizaje por inducción," *Reportes Técnicos en Ingeniería del Software*, vol. 6, no. 2, pp. 64–69, 2004.

[5] Instituto Nacional de Estadística, "Población que usa Internet (en los últimos tres meses)," http://www.ine.es/ss/Satellite?L=es_ES&c=INESeccion_C&cid=1259925528782&p=1254735110672&pagename=ProductosYServicios%2FPYSLayout, 2016.

[6] Fundación Telefónica, "La Sociedad de la Información en España 2016," https://www.fundaciontelefonica.com/arte_cultura/publicaciones-listado/pagina-item-publicaciones/itempubli/558/, 2017.

[7] Boletín Oficial del Estado, "Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal." https://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750, pp. 43 088–43 099, 14/12/1999.

[8] ——, "Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos)." https://www.boe.es/diario_boe/txt.php?id=DOUE-L-2016-80807, pp. 1–88, 04/05/2016.

[9] A. Frank and A. Asuncion, "UCI machine learning repository," 2010, university of California, Irvine, School of Information and Computer Sciences. [Online]. Available: http://archive.ics.uci.edu/ml

[10] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998. [Online]. Available: http://dx.doi.org/10.1109/34.667881

[11] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," *Ann. Statist.*, vol. 26, no. 5, pp. 1651–1686, 10 1998. [Online]. Available: http://dx.doi.org/10.1214/aos/1024691352

[12] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996. [Online]. Available: http://dx.doi.org/10.1023/A:1018054314350

[13] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002200009791504X

[14] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[15] I. Cantador and J. R. Dorronsoro, *Balanced Boosting with Parallel Perceptrons*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 208–216. [Online]. Available: http://dx.doi.org/10.1007/11494669_26

[16] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, vol. 137, no. 1, pp. 239 – 263, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000437020200190X

[17] N. Shigei and H. Miyajima, "Bagging and boosting algorithms for support vector machine classifiers," in *Proceedings of the 8th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, ser. AIKED'09. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 372–377. [Online]. Available: http://dl.acm.org/citation.cfm?id=1553921.1553991

[18] G. Valentini and T. G. Dietterich, "Bias-variance analysis of support vector machines for the development of svm-based ensemble methods," *J. Mach. Learn. Res.*, vol. 5, pp. 725–775, Dec. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1005332.1016783

[19] L. Breiman, "Randomizing outputs to increase prediction accuracy," *Machine Learning*, vol. 40, no. 3, pp. 229–242, 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1007682208299

[20] G. Martínez-Muñoz and A. Suárez, "Switching class labels to generate classification ensembles," *Pattern Recognition*, vol. 38, pp. 1483–1494, 2005.

[21] G. Martínez-Muñoz, A. Sánchez-Martínez, D. Hernández-Lobato, and A. Suárez, "Class-switching neural network ensembles," *Neurocomputing*, vol. 71, no. 13?15, pp. 2521–2528, 2008.

[22] MATLAB, "Statistics and Machine Learning Toolbox," https://es.mathworks.com/help/stats/, 2017.

[23] ——, "Neural Network Toolbox," https://es.mathworks.com/help/nnet/, 2017.

[24] UNE-ISO, "Directrices para la dirección y gestión de proyectos." http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0050883#.WUOjn2jyjIV, 2013-03-20.